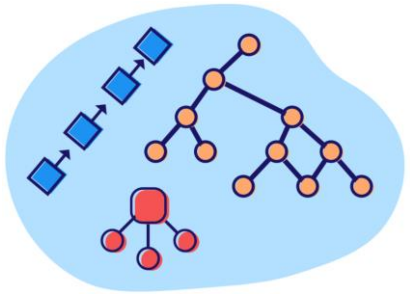
الجامعة العربية الخاصة للعلوم والتكنولوجيا
كلية الهندسة المعلوماتية
قسم تقانة المعلومات

Algorithms & Data Structures (2)

Lecture-7: Binary Search Trees

(أشجار البحث الثنائية)



Dr. Asmaa Shaar

فهرس المحاضرة

- تعريف شجرة البحث الثنائية
- تمثيل أشجار البحث الثنائي
- البحث في شجرة البحث الثنائية
- البحث عن أصغر key في شجرة البحث الثنائية
- البحث عن أكبر key في شجرة البحث الثنائية
- الإضافة في شجرة البحث الثنائية
- بناء شجرة بحث ثنائية
- الحذف في شجرة البحث الثنائية

أنواع الاشجار

- Binary tree
  - Binary search tree, AVL tree, Red black tree,…
- B tree
- Heaps
- Application Specific tree
- …

تعريف شجرة البحث الثنائية

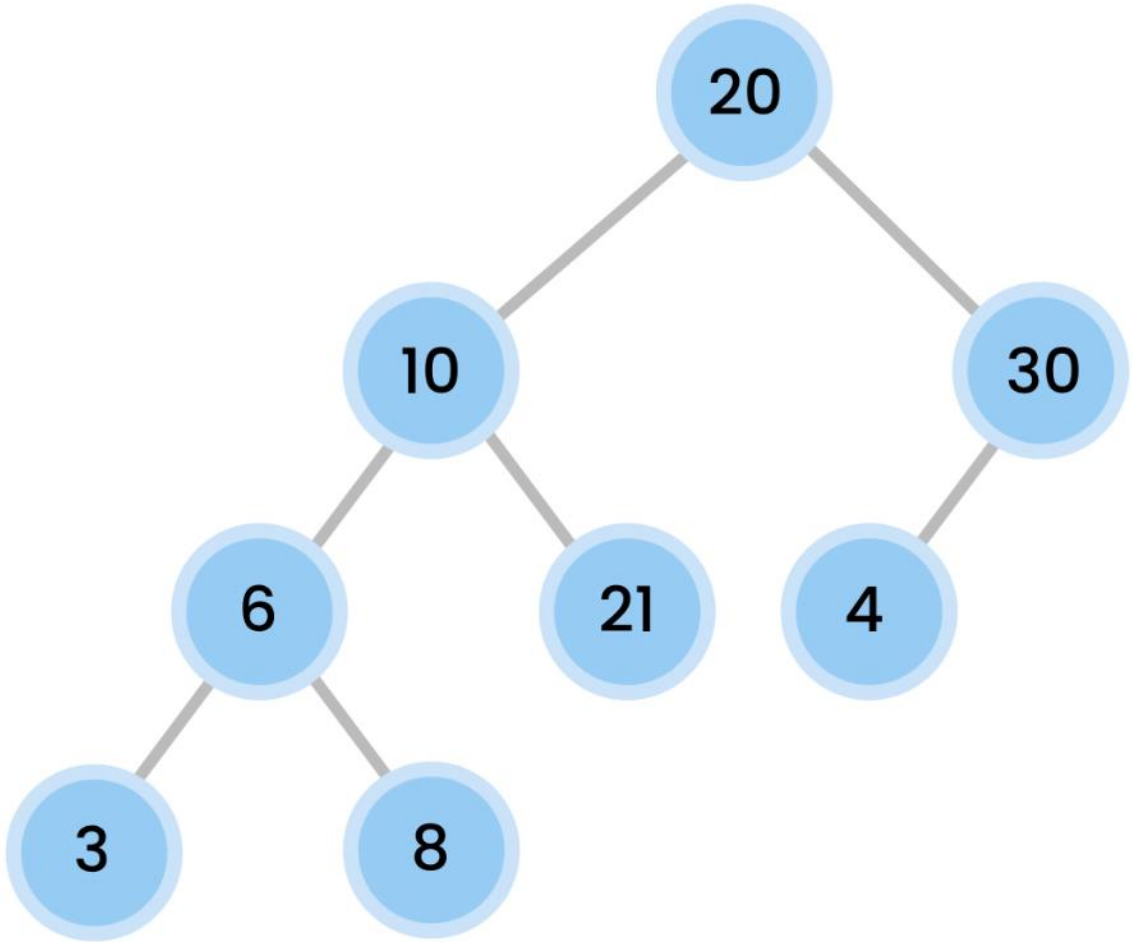
- نقول عن شجرة ثنائية B إنها **شجرة بحث ثنائية** إذا تحقق أحد الشرطين:
  - B فارغة
  - إذا كانت B غير فارغة ، يجب تحقق الشروط التالية:

1. لكل عقدة في الشجرة **قيمة مفتاحية** (key value)، ولا يمكن أن تكون هناك عقد متشابهة في القيمة المفتاحية (لا تتضمن **قيماً مكررة**).
2. كل القيم المفتاحية للعقد في الشجرة الفرعية **اليسارية أصغر من** قيمة الجذر.
3. كل القيم المفتاحية للعقد في الشجرة الفرعية **اليمينية أكبر من** قيمة الجذر.
4. كل من الشجرة الفرعية اليسارية واليمينية هي **شجرة بحث ثنائية** أيضاً.

- **ملاحظة**: يمكن أن توضع القيم الصغيرة على الطرف الأيمن والقيم الكبيرة على الطرف الأيسر.

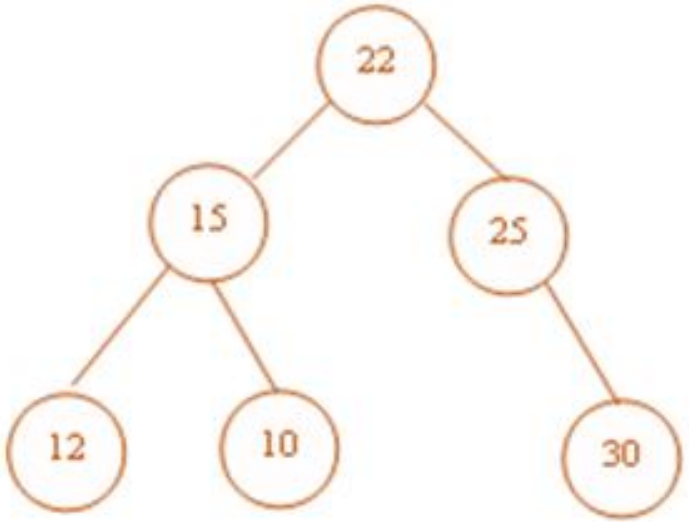
أشجار البحث الثنائي

- مثال : الشجرة التالية ليست شجرة بحث ثنائية .

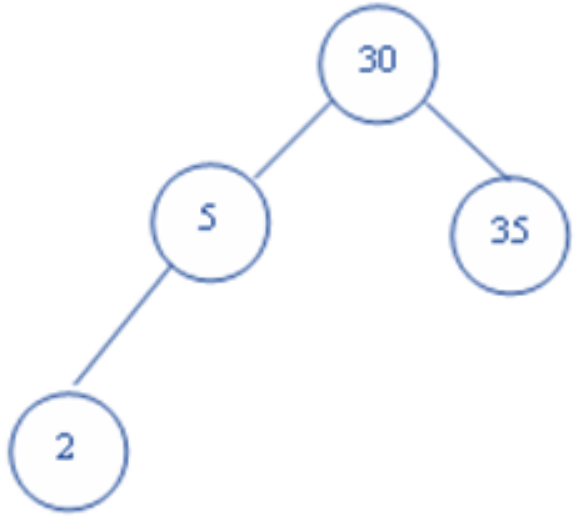


أشجار البحث الثنائي

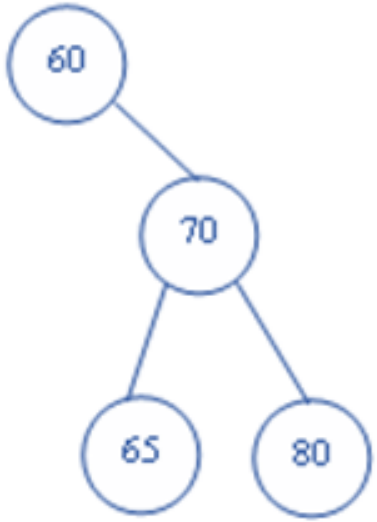
- مثال : كل من الشجرتين (c) ,(b) شجرة بحث ثنائية أما الشجرة (a) فهي ليست شجرة بحث ثنائية (لم تحقق الشرط الرابع) .



(a)



(b)



(c)

أشجار البحث الثنائي

- تفرض الخصائص السابقة **ترتيباً خاصاً للعناصر** في شجرة البحث الثنائية يعتمد على **قيمة المفتاح**، ونتيجة لذلك **تزداد سرعة تنفيذ** العمليات التالية:
  - البحث
  - الإضافة
  - الحذف

- ولولا وجود هذا الترتيب كان يجب علينا مقارنة جميع المفاتيح مع المفتاح المعطى للحصول على النتيجة المطلوبة.

تمثيل أشجار البحث الثنائي

- بما أن شجرة البحث الثنائية هي **شكل خاص** من الشجرة الثنائية وبالتالي تمثيل (تحقيق) شجرة البحث الثنائية **لا يختلف** عن التمثيل المستخدم من أجل الشجرة الثنائية.
  - باستخدام المصفوفات
  - باستخدام القوائم المترابطة

✓ Key: لتخزين المحتوى الفعلي للعقدة (القيم المفتاحية).

✓ left: مؤشر إلى الشجرة الفرعية اليسارية

✓ right:مؤشر إلى الشجرة الفرعية اليمينية

```
struct node {
        int key;
        node *left, *right;
                        };
node *root;          // تكون الشجرة عبارة عن مؤشر إلى العقدة الأولى ( أي الجذر)
```

البحث في شجرة البحث الثنائية

- تبدأ عملية البحث عن قيمة مفتاحية معينة x في شجرة البحث الثنائية **بمقارنة هذه القيمة المفتاحية مع الجذر**.
  - إذا كان x **يساوي** القيمة المفتاحية للجذر ← **تم إيجاد العنصر المطلوب** البحث عنه ونعود بمؤشر على الجذر.
  - إذا كان x **أصغر** من القيمة المفتاحية للجذر ← **يتم البحث عن x** في الشجرة الفرعية اليسارية للجذر.
  - إذا كان x **أكبر** من القيمة المفتاحية للجذر ← **يتم البحث عن x** في الشجرة الفرعية اليمينية للجذر.
  - إذا كان الجذر يساوي NULL ← شجرة البحث لا تضمن أي عنصر وبالتالي x **غير موجود** لذلك نعود بـ null.

البحث في شجرة البحث الثنائية

- الدالة **الإجرائية العودية** المسؤولة عن تنفيذ عملية البحث هي:

```
node* search(node* root, int x)
      {
          // Base Cases: root is null or x is present at root
           if (root == Null || x == root->key )
              return root;
          // x is smaller than root's key
           if (x < root->key)
                 return search(root->left, x);
           else return search (root -> right, x);
             }
```

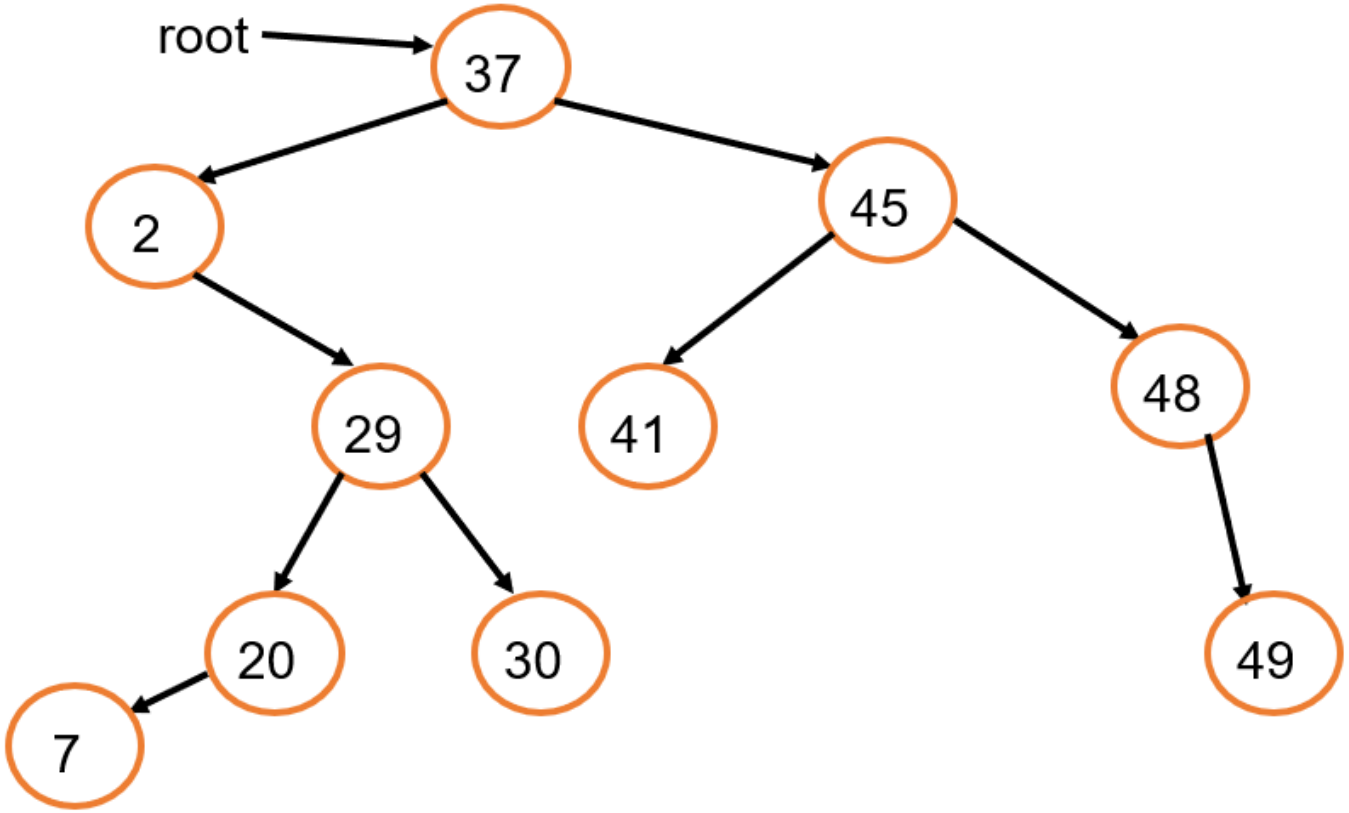
البحث في شجرة البحث الثنائية

- الدالة **الإجرائية التكرارية** المسؤولة عن تنفيذ عملية البحث هي:

```
node* iter_search(node* root, int x)
  {
        while (root !=Null)
            {
          if (x == root->key ) return root;
          if (x < root->key)
             root = root->left;
          else
             root = root->right;
              }
         return Null;
               }
```

البحث في شجرة البحث الثنائية

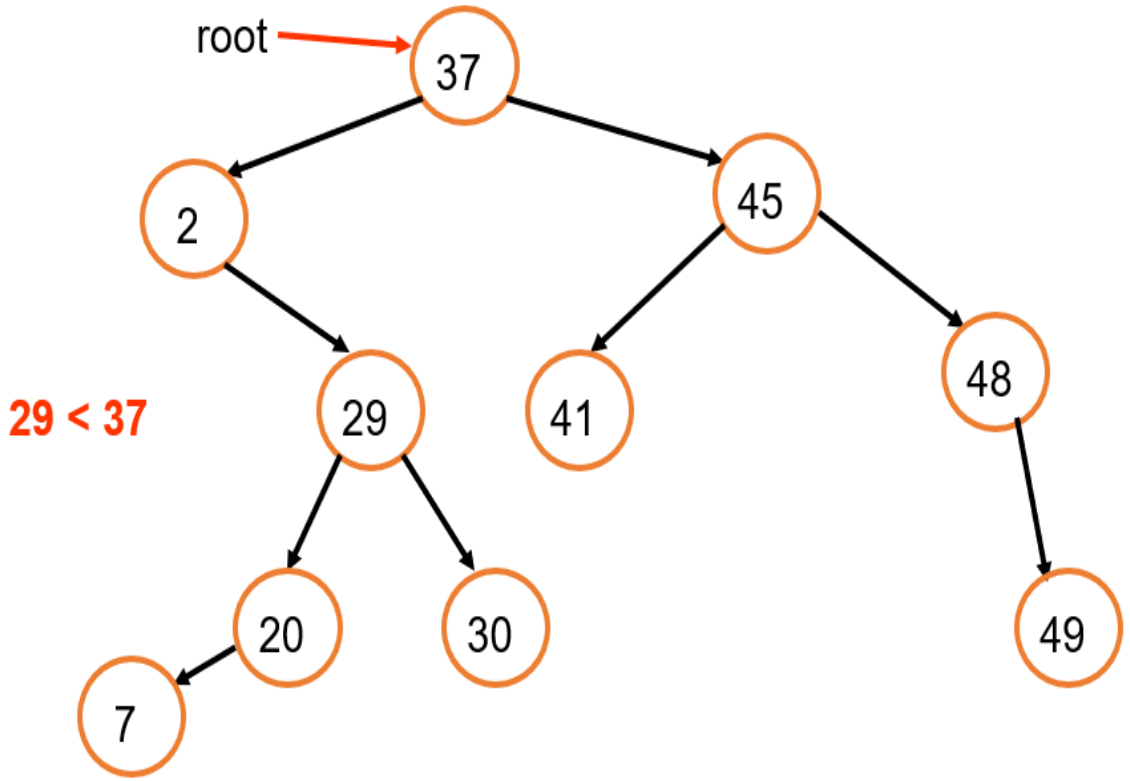
- مثال1:

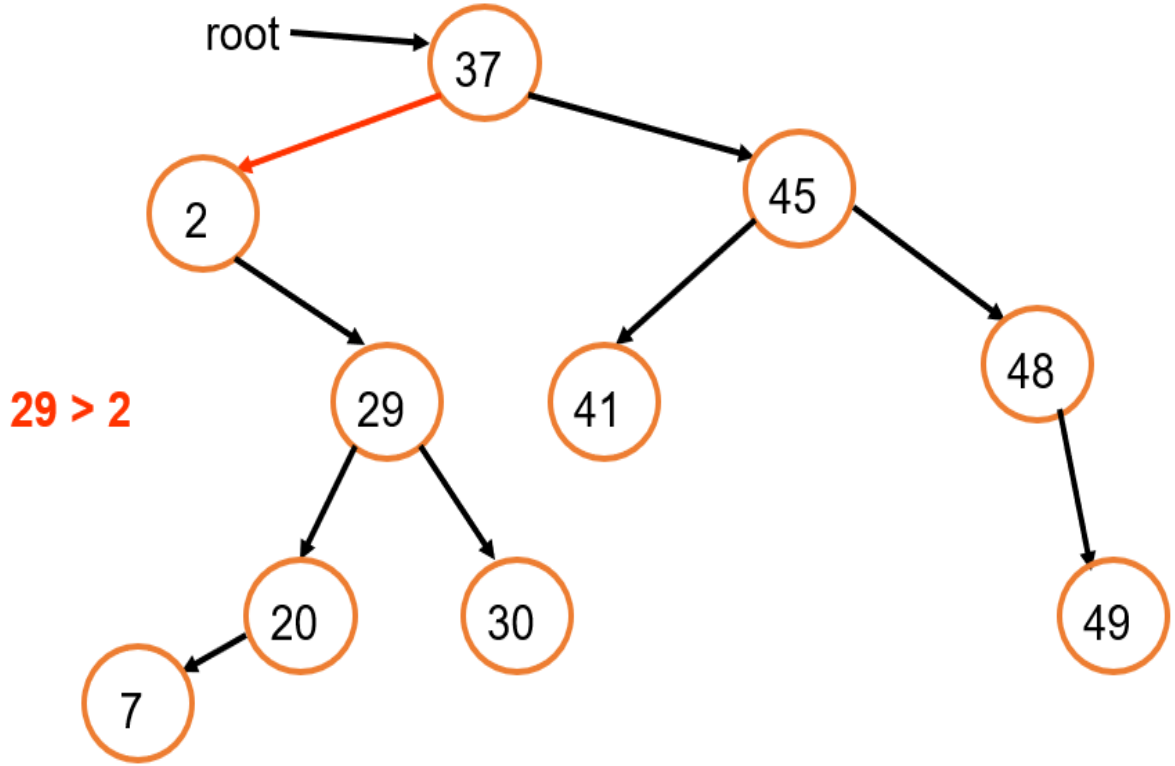


Key to search for: 29

البحث في شجرة البحث الثنائية

- مثال1:

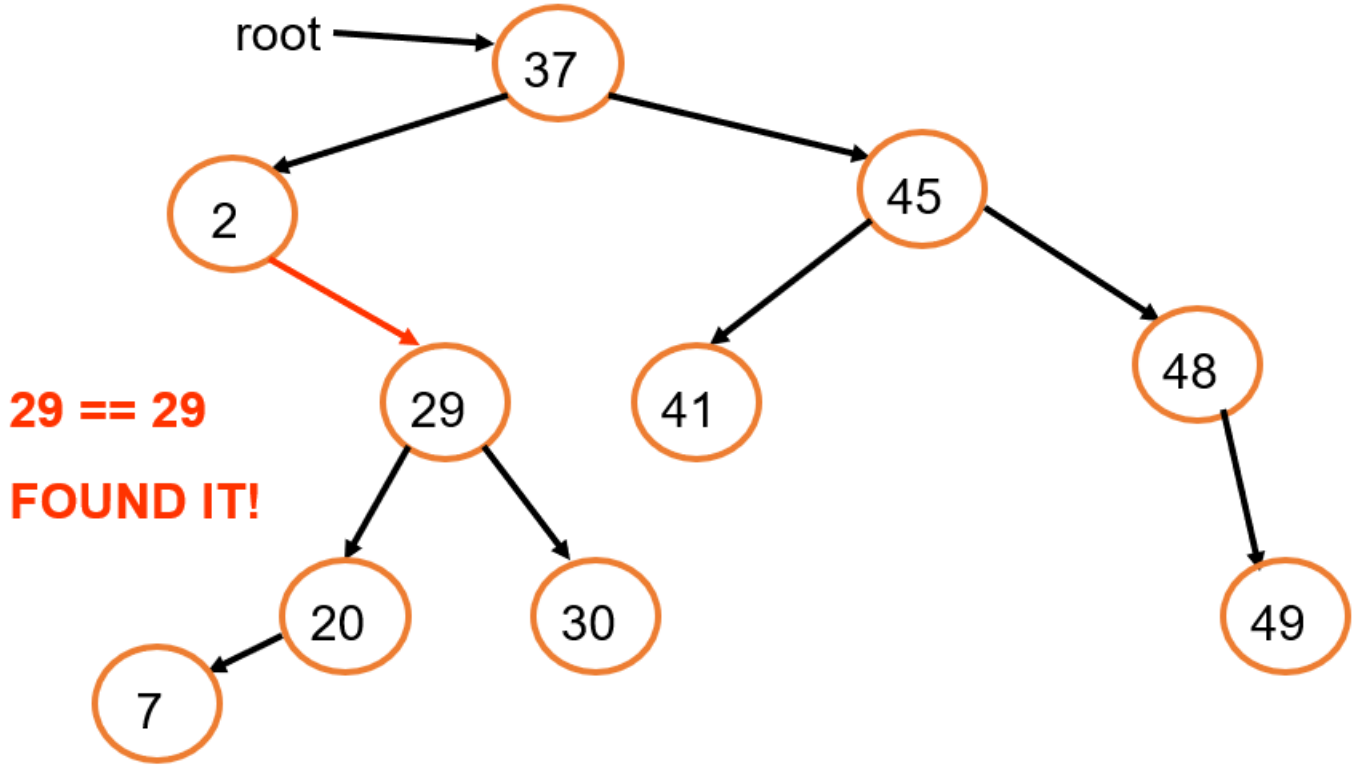




Key to search for: 29

البحث في شجرة البحث الثنائية

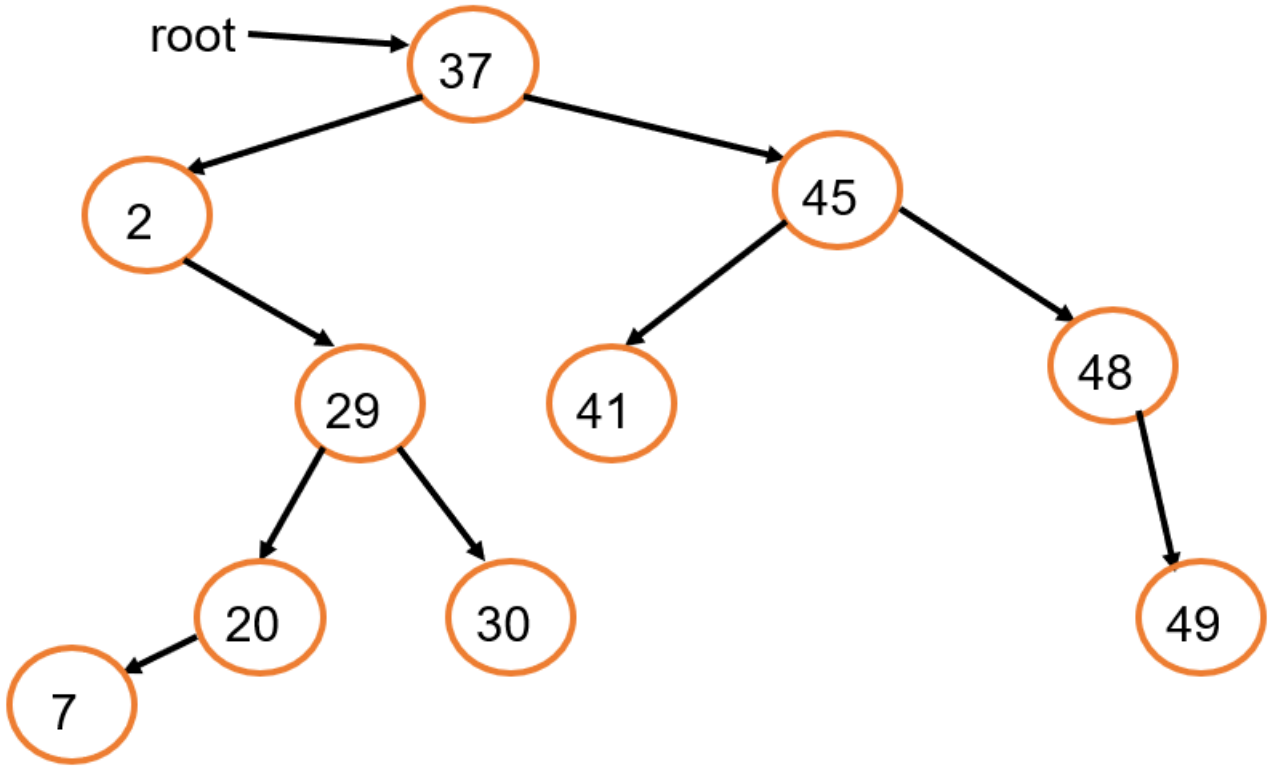
- مثال1:



Key to search for: 29

البحث في شجرة البحث الثنائية

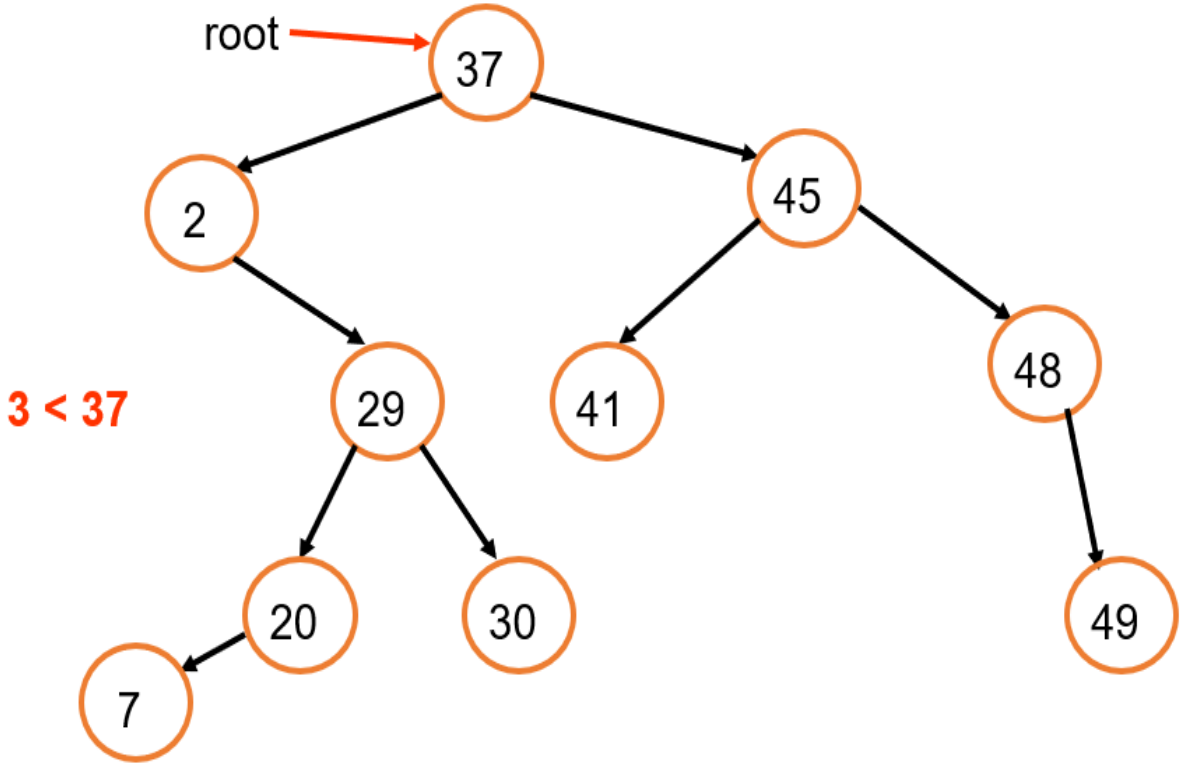
- مثال2:

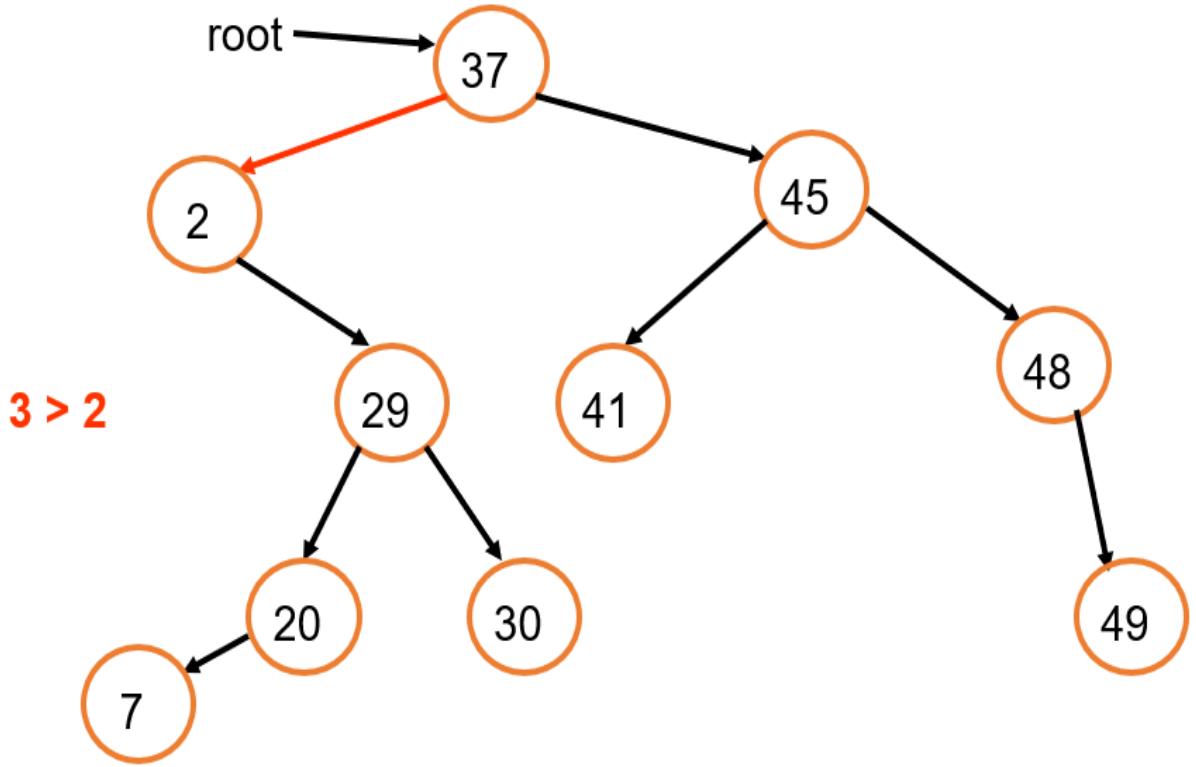


Key to search for: 3

البحث في شجرة البحث الثنائية

- مثال2:

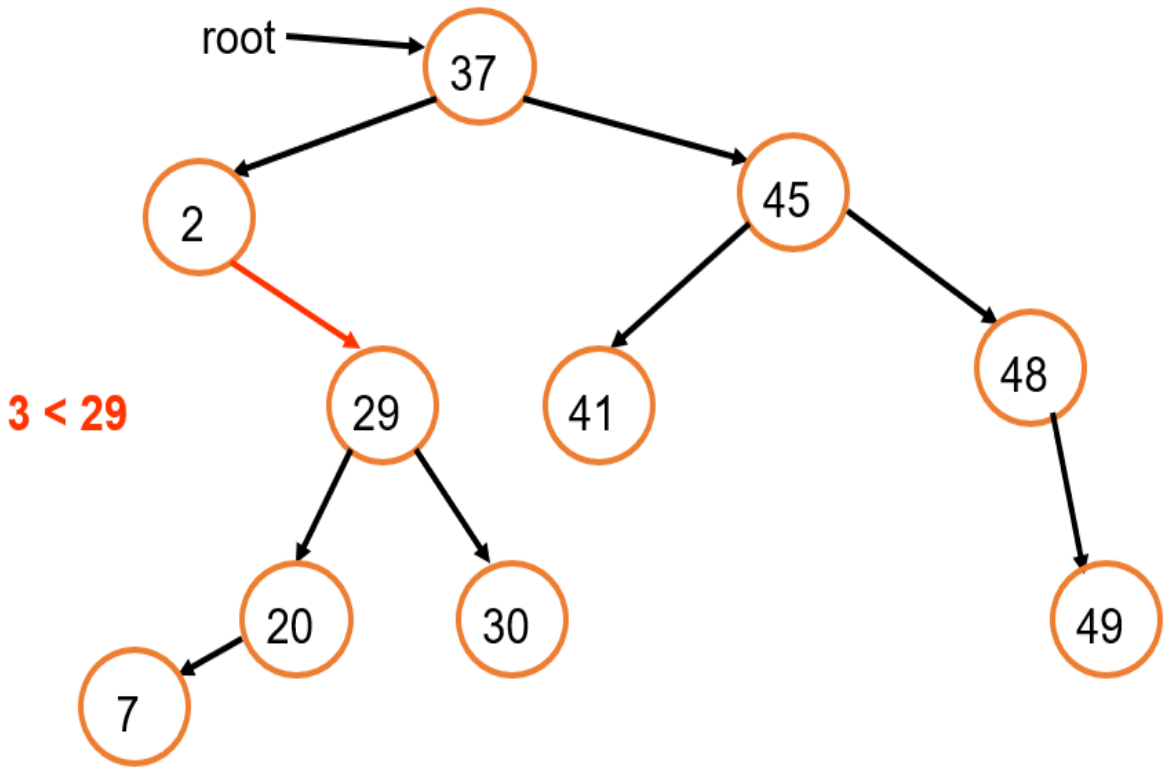


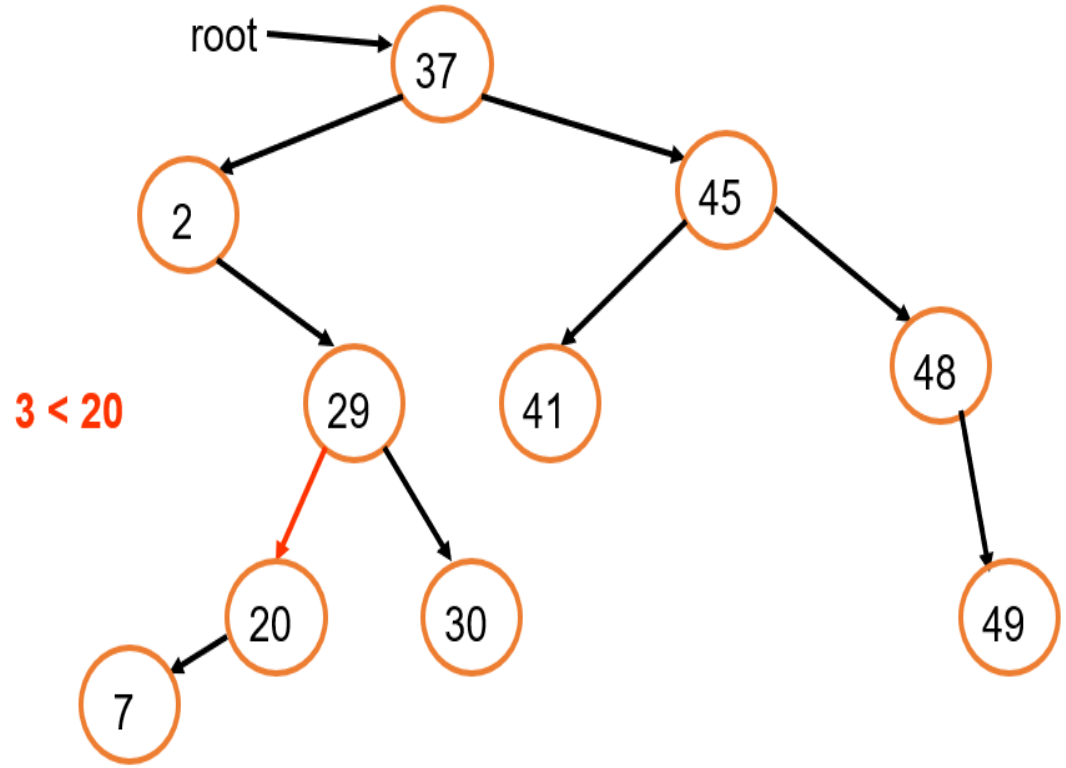


Key to search for: 3

البحث في شجرة البحث الثنائية

- مثال2:

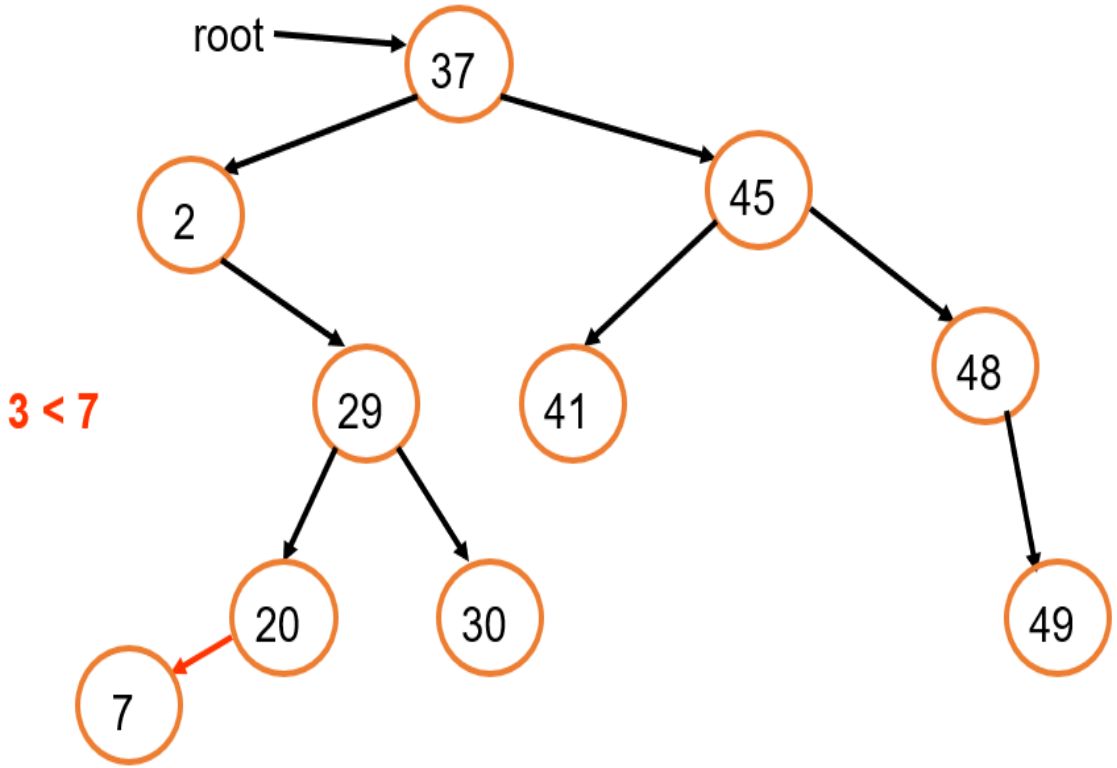


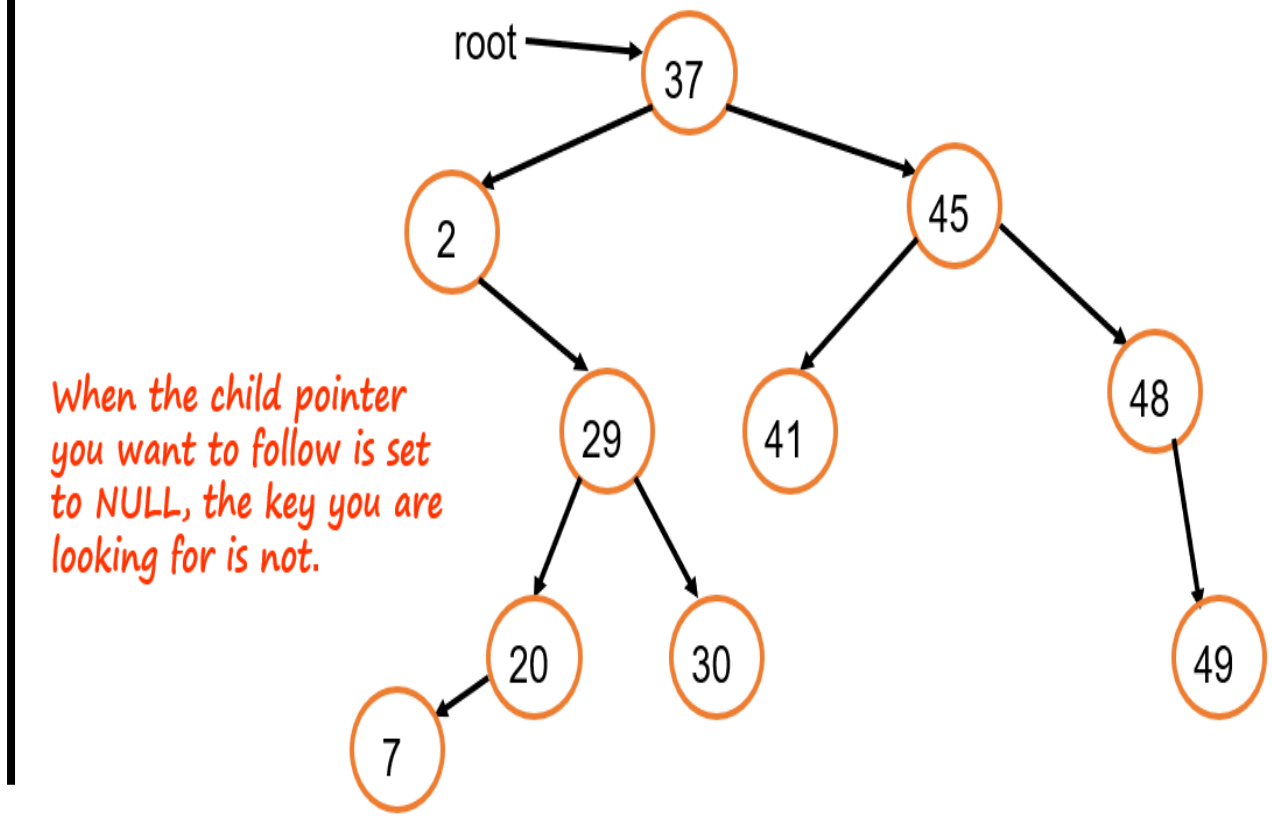


Key to search for: 3

البحث في شجرة البحث الثنائية

- مثال2:





Key to search for: 3

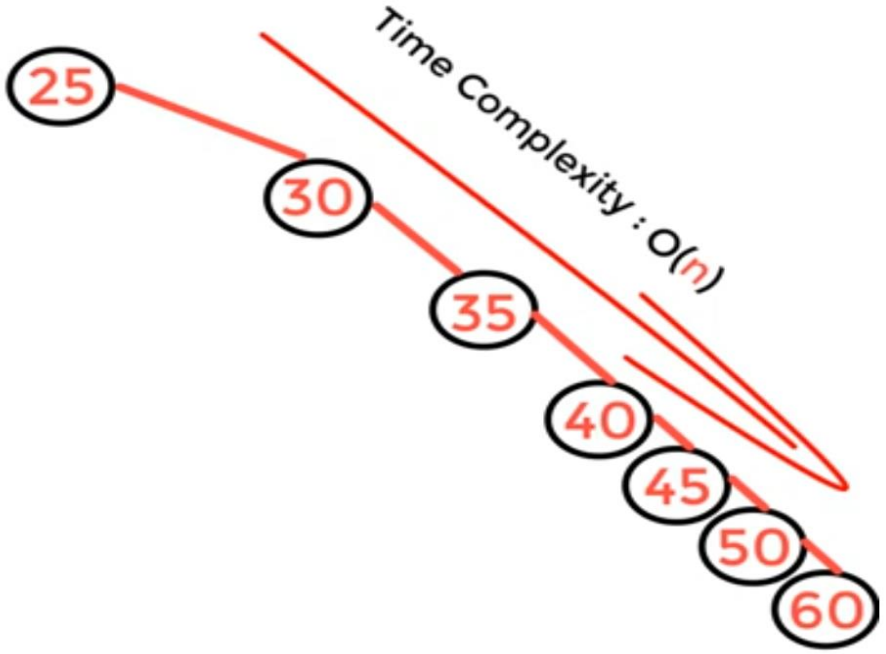
تحليل دالة البحث

- يتم إنجاز البحث باستخدام الدالة search() أو iter_search() بزمن قدره O(h) حيث h ارتفاع الشجرة :
- **أفضل حالة:** شجرة البحث الثنائية هي **شجرة ثنائية كاملة**، عندئذ:

Time Complexity : $O(\log_2 n)$

$\log_2(10): 4$
$\log_2(1000): 10$
$\log_2(1000000): 20$
$\log_2(1000000000): 30$

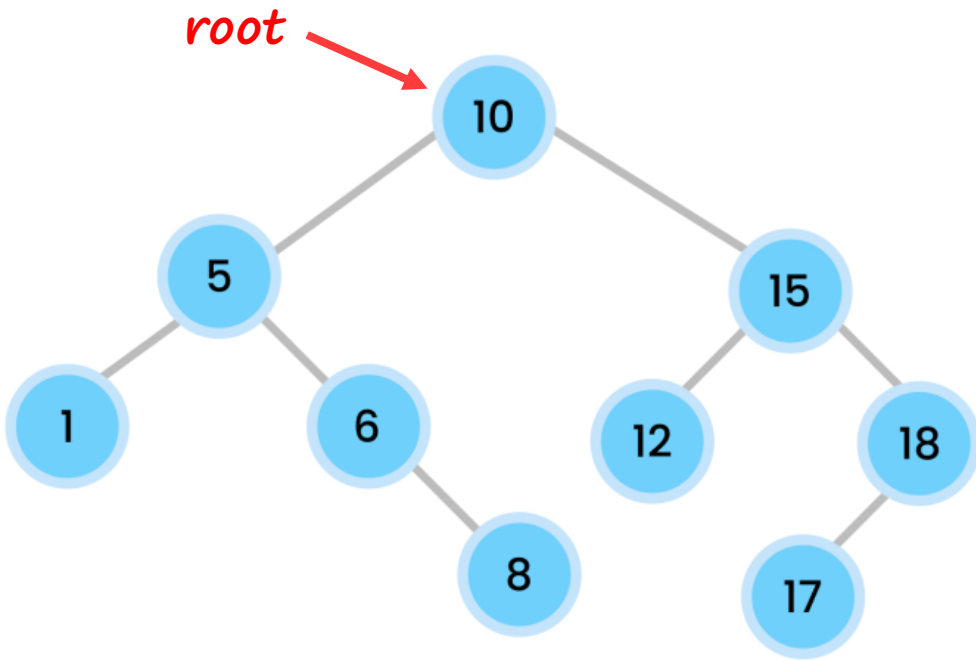
- **أسوأ حالة:** شجرة البحث الثنائية هي **شجرة خطية**، عندئذ:



البحث عن أصغر key في شجرة البحث الثنائية

- يبدأ البحث بالذهاب إلى **الابن اليساري** لعقدة الجذر، ومن ثم إلى **ابنه اليساري**، وهكذا حتى يتم الوصول إلى عقدة **لا تملك ابناً يسارياً** عندئذ تكون هذه العقدة تملك **أصغر قيمة مفتاحية** في الشجرة.
- الدالة الإجرائية التكرارية المسؤولة:

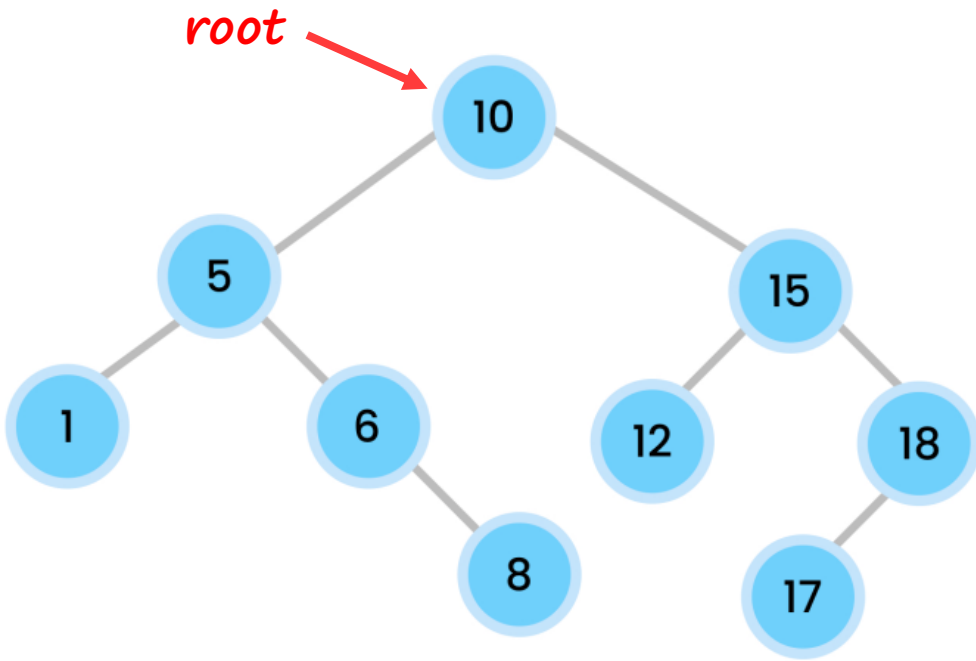
```
node* minimum (node* root)
    {
       node* ptr;
       while(root !=null){
              ptr = root;
              root= root ->left;
                          }
       return ptr;
    }
```



البحث عن أكبر key في شجرة البحث الثنائية

- يبدأ البحث بالذهاب إلى **الابنِ اليمينِي** لعقدة الجذر، ومن ثم إلى **ابنه اليميني**، وهكذا حتى يتم الوصول إلى عقدة **لا تملك ابناً يمينياً** عندئذ تكون هذه العقدة تملك **أكبر قيمة مفتاحية** في الشجرة.
- الدالة الإجرائية التكرارية المسؤولة:

```
node* maximum (node* root)
    {
      node* ptr;
      while(root !=null){
            ptr = root;
            root= root->right;
                        }
       return ptr;
     }
```



الإضافة في شجرة البحث الثنائية

- لإضافة عقدة جديدة بقيمة مفتاحية x ، يجب أن نتأكد أولاً من أن x مختلفة عن أي key في الشجرة.
- يتم تحقيق ما سبق باستخدام دالة البحث modified_search() (نسخة معدلة عن iter_search()) على شجرة **ليست فارغة** والتي بدورها:
  - تعود بـ null إذا كانت الشجرة تتضمن عقدة مع قيمة مفتاحية تساوي x .
  - وغير ذلك، تعود بمؤشر آخر عقدة وصل إليها البحث. ← يدرج العنصر الجديد كأبن يساري أو يميني لهذه العقدة.

```
node* modified_search(node* root, int x)
  {       node* ptr=Null;
          while (root !=Null) {
            ptr = root;
            if (x == root->key) return Null;
            if (x < root ->key)
                  root = root->left;
            else  root = root->right;}
            return ptr ;}
```

الإضافة في شجرة البحث الثنائية

- يتم الإضافة في شجرة البحث الثنائية باستخدام دالة إجرائية insert_node:
  - في البداية لدينا **newNode** دالة مساعدة لإنشاء عقدة جديدة في شجرة البحث الثنائية:

```
node*   newNode(int x)
      { node* temp = (node *)malloc(sizeof(node));
        temp->key = x;
        temp->left = temp->right = NULL;
        return temp;
      }
```

الإضافة في شجرة البحث الثنائية

- يتم الإضافة في شجرة البحث الثنائية باستخدام دالة إجرائية insert_node :

```
void insert_node (node* root, int x)
          {
            if (root == null)
               {root=newNode(x);
                 return;}

           node* temp = modified_search (root, x);
           if (temp != null){          // x is not in the tree
                if (x < temp-> key)
                   temp->left = newNode(x);
                else
                   temp -> right = newNode(x);
                              }

           }
```

تحتاج الدالة insert_node إلى زمن حساب قدره O(log(n)) لتنفيذ البحث على الشجرة، و كذلك تحتاج إلى O(1) من الزمن لإضافة عقدة في الموقع المناسب من الشجرة وبالتالي الزمن الكلي للدالة insert_node هو O(log(n)) .

بناء شجرة بحث ثنائية

- **مثال1:** المطلوب بناء شجرة البحث الثنائية لقائمة العناصر التالية.

37, 2, 45, 48, 41, 29, 20, 30, 49, 7

بناء شجرة بحث ثنائية

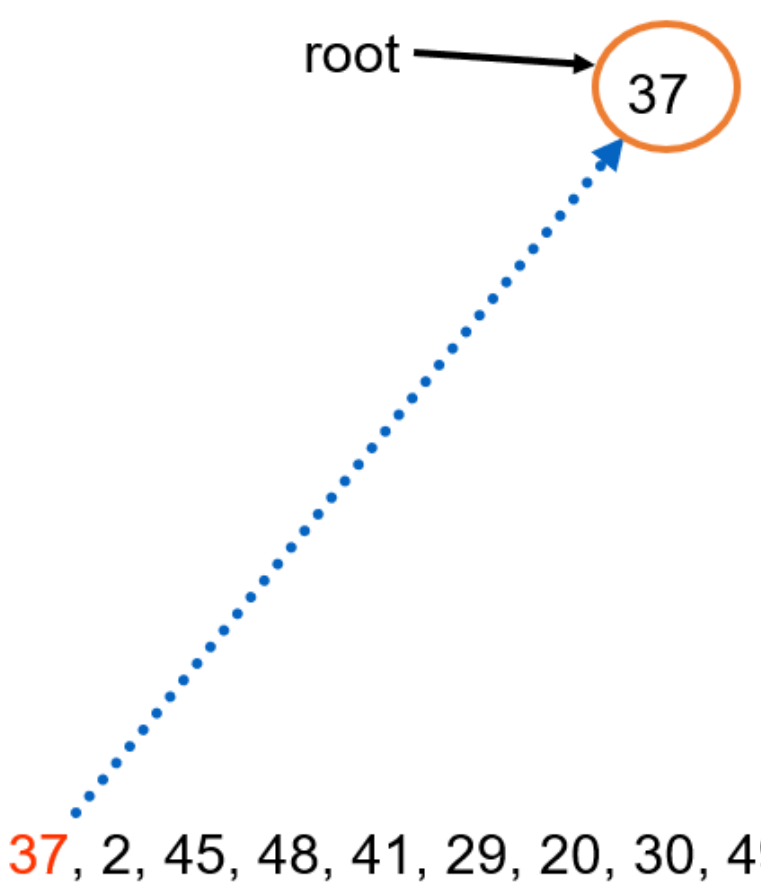
- **مثال1:** المطلوب بناء شجرة البحث الثنائية لقائمة العناصر التالية.

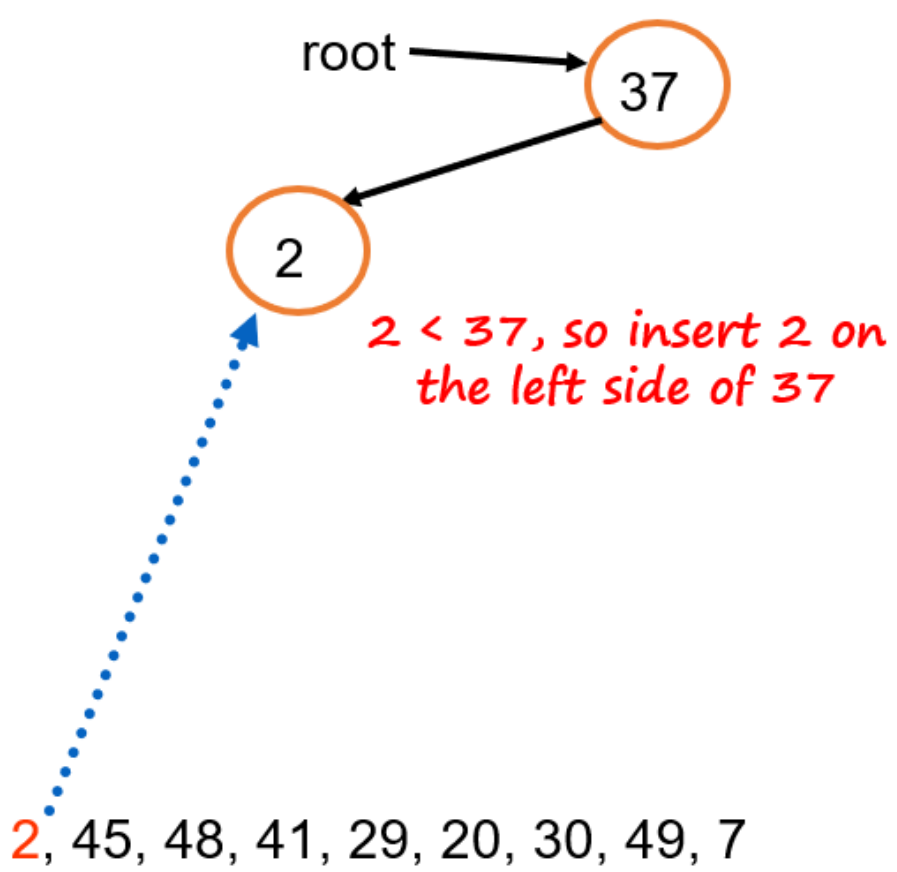
root: NULL

37, 2, 45, 48, 41, 29, 20, 30, 49, 7

بناء شجرة بحث ثنائية

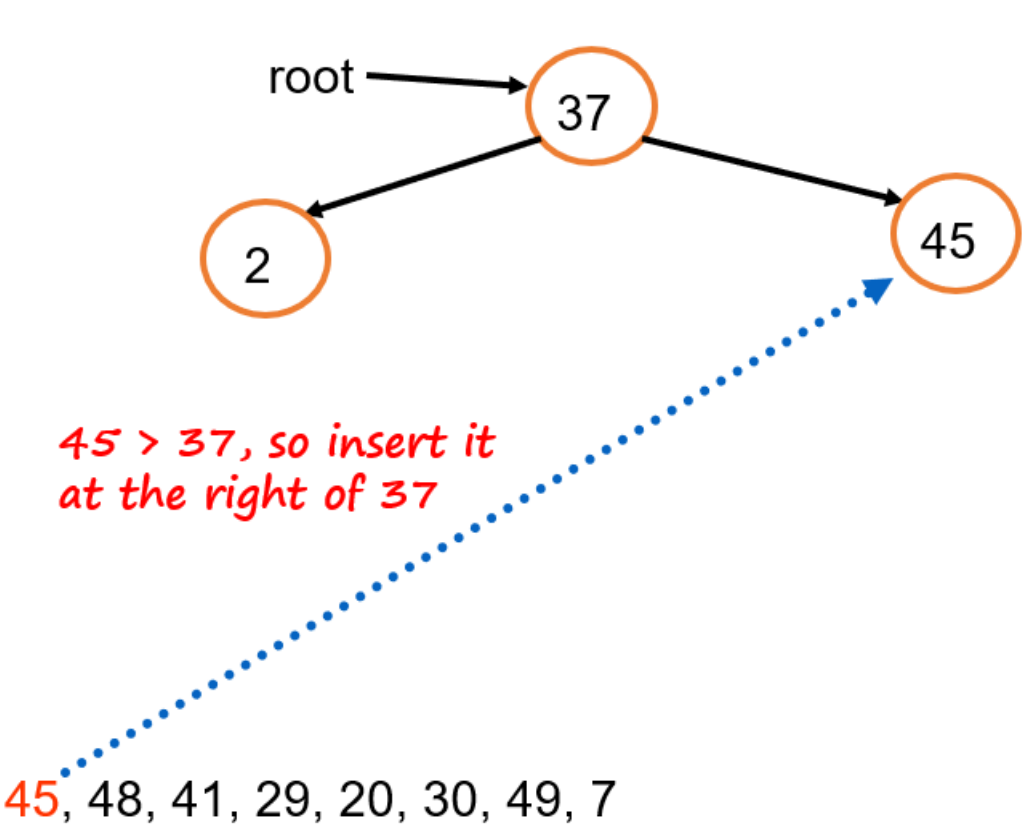
- **مثال1:** المطلوب بناء شجرة البحث الثنائية الموافقة لقائمة العناصر التالية.

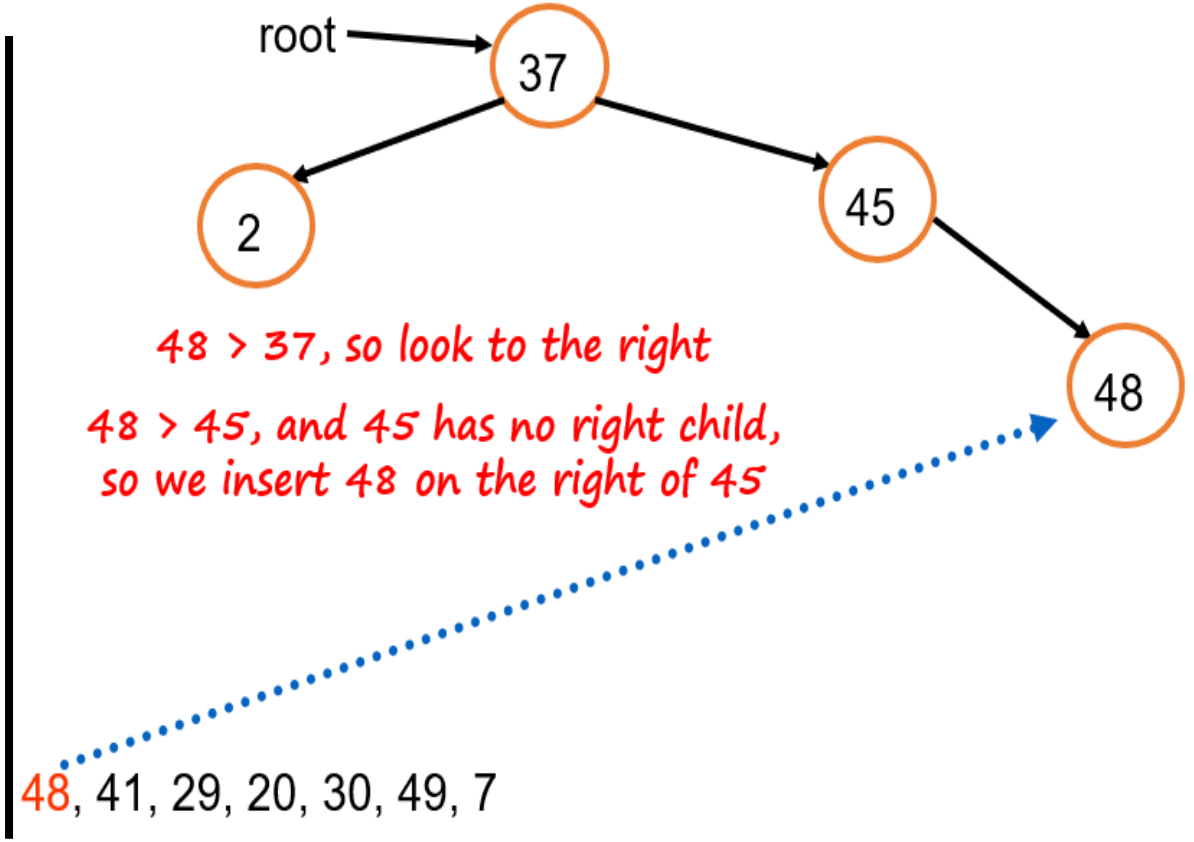




بناء شجرة بحث ثنائية

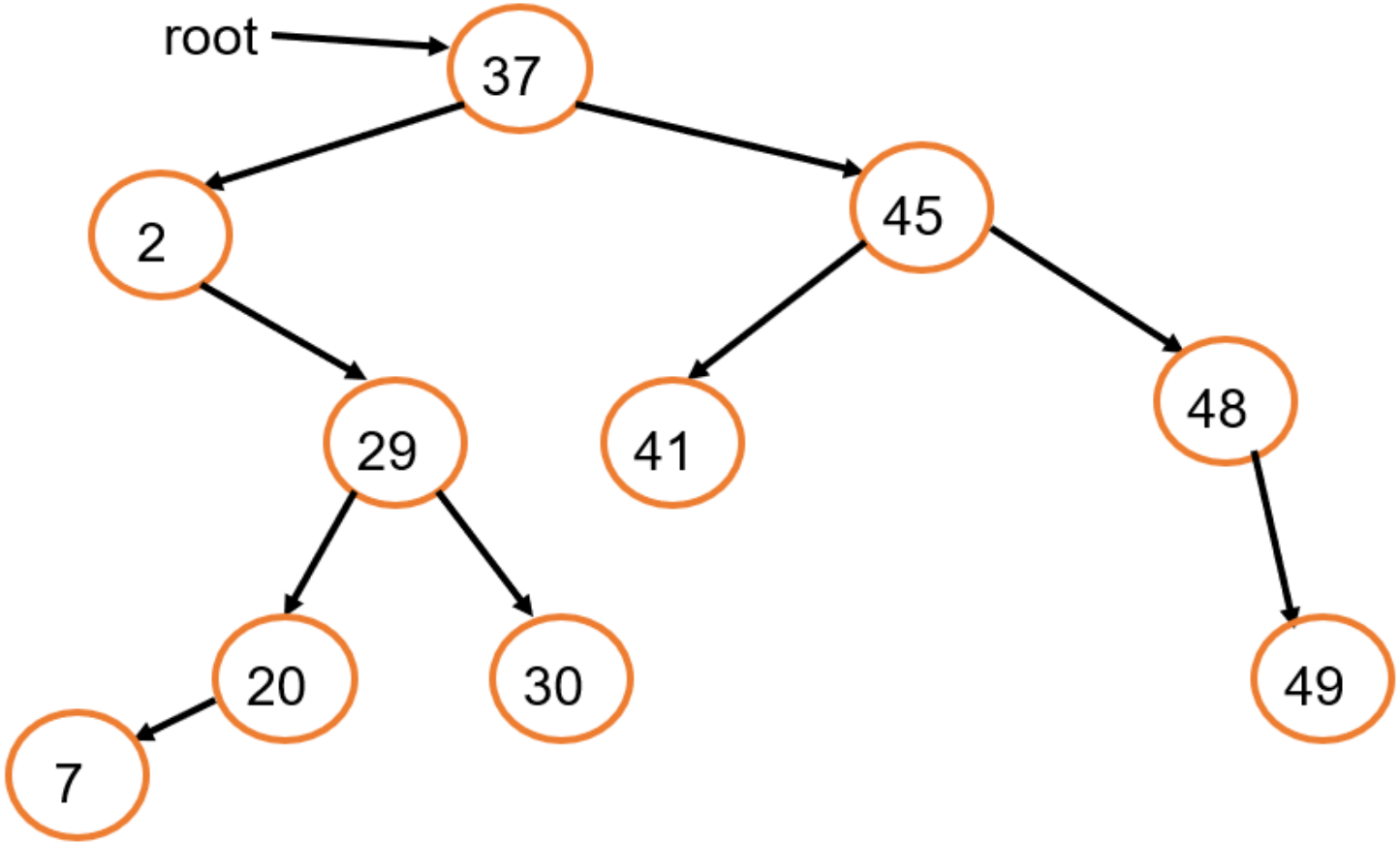
- **مثال1:** المطلوب بناء شجرة البحث الثنائية الموافقة لقائمة العناصر التالية.





بناء شجرة بحث ثنائية

- **مثال1:** المطلوب بناء شجرة البحث الثنائية الموافقة لقائمة العناصر التالية.



All elements have been inserted

بناء شجرة بحث ثنائية

- **مثال2:** المطلوب بناء شجرة البحث الثنائية لقائمة العناصر التالية.

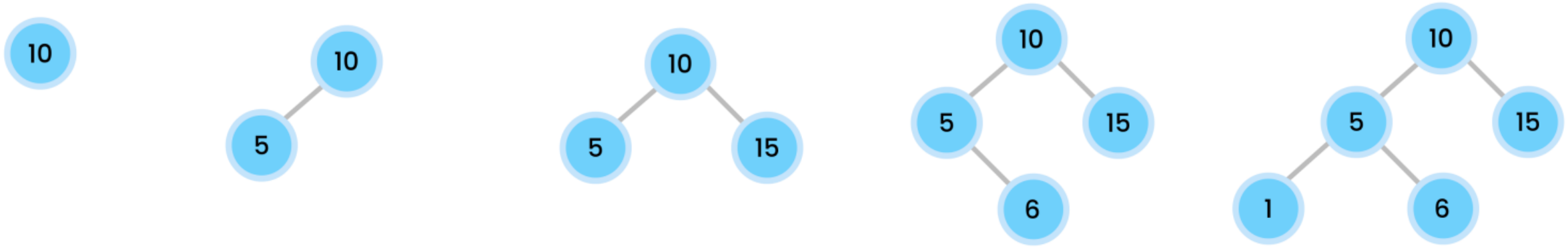
10, 5, 15, 6, 1, 8, 12, 18, 17

You can also use the following tool to see how a binary search tree evolves as you add new items:
https://visualgo.net/en/bst

بناء شجرة بحث ثنائية

- **مثال2:** المطلوب بناء شجرة البحث الثنائية لقائمة العناصر التالية.

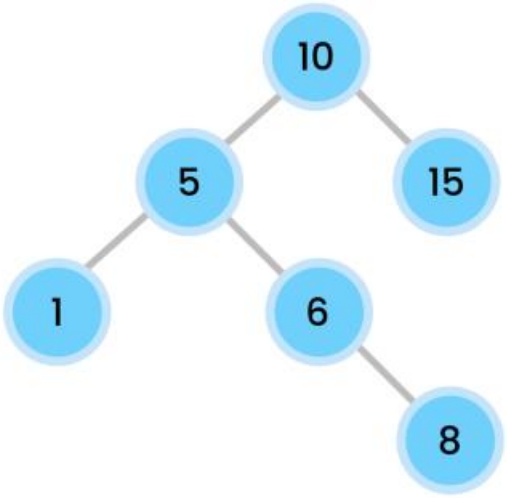
10, 5, 15, 6, 1, 8, 12, 18, 17

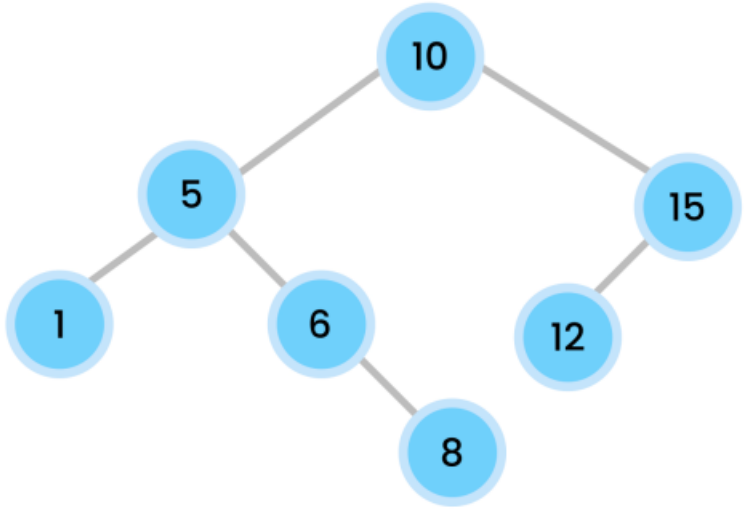


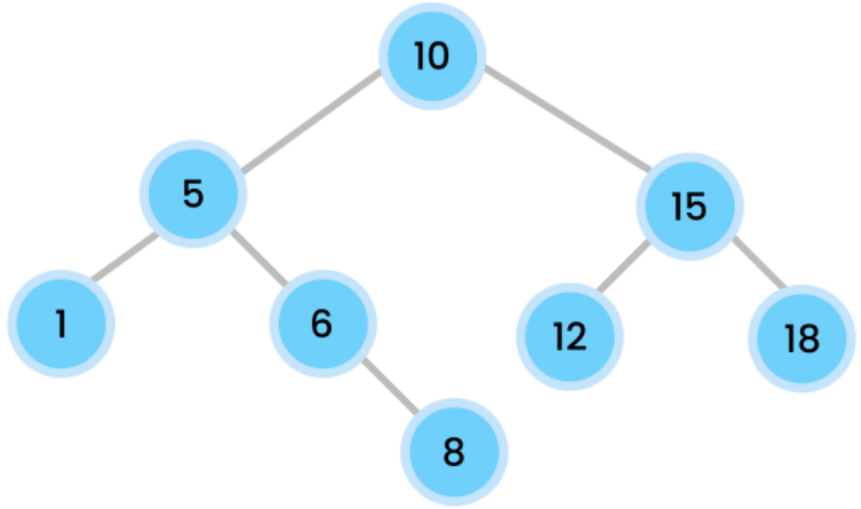
بناء شجرة بحث ثنائية

- **مثال2:** المطلوب بناء شجرة البحث الثنائية لقائمة العناصر التالية.

10, 5, 15, 6, 1, 8, 12, 18, 17



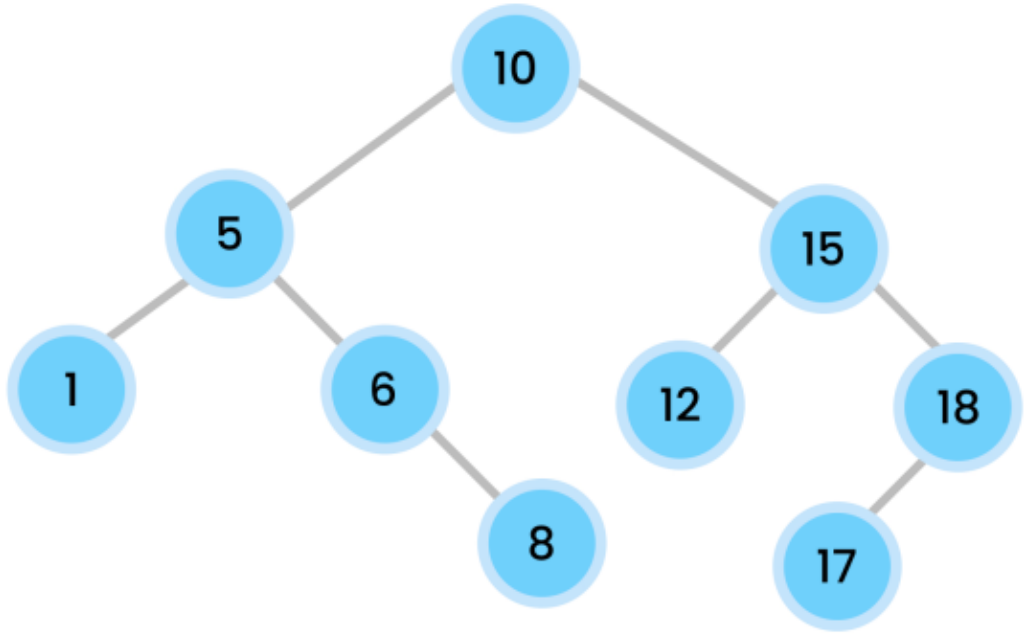




بناء شجرة بحث ثنائية

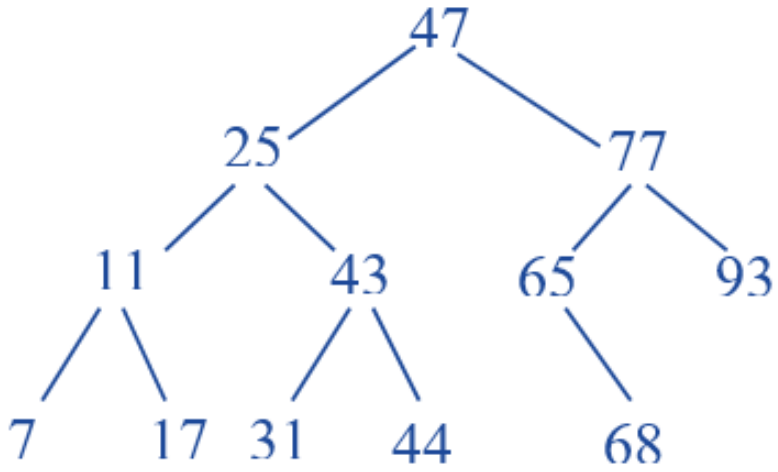
- **مثال2:** المطلوب بناء شجرة البحث الثنائية لقائمة العناصر التالية.

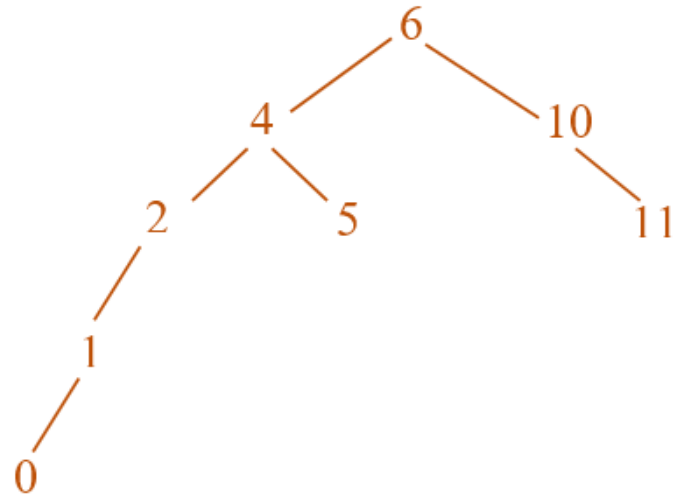
10, 5, 15, 6, 1, 8, 12, 18, 17



بناء شجرة بحث ثنائية

- **مثال 3** : المطلوب بناء شجرة البحث الثنائية لقائمة العناصر التالية:
{ 47 , 25 , 77 , 11 , 43 , 93 , 65 , 31 , 17 , 7 , 44 , 68 }



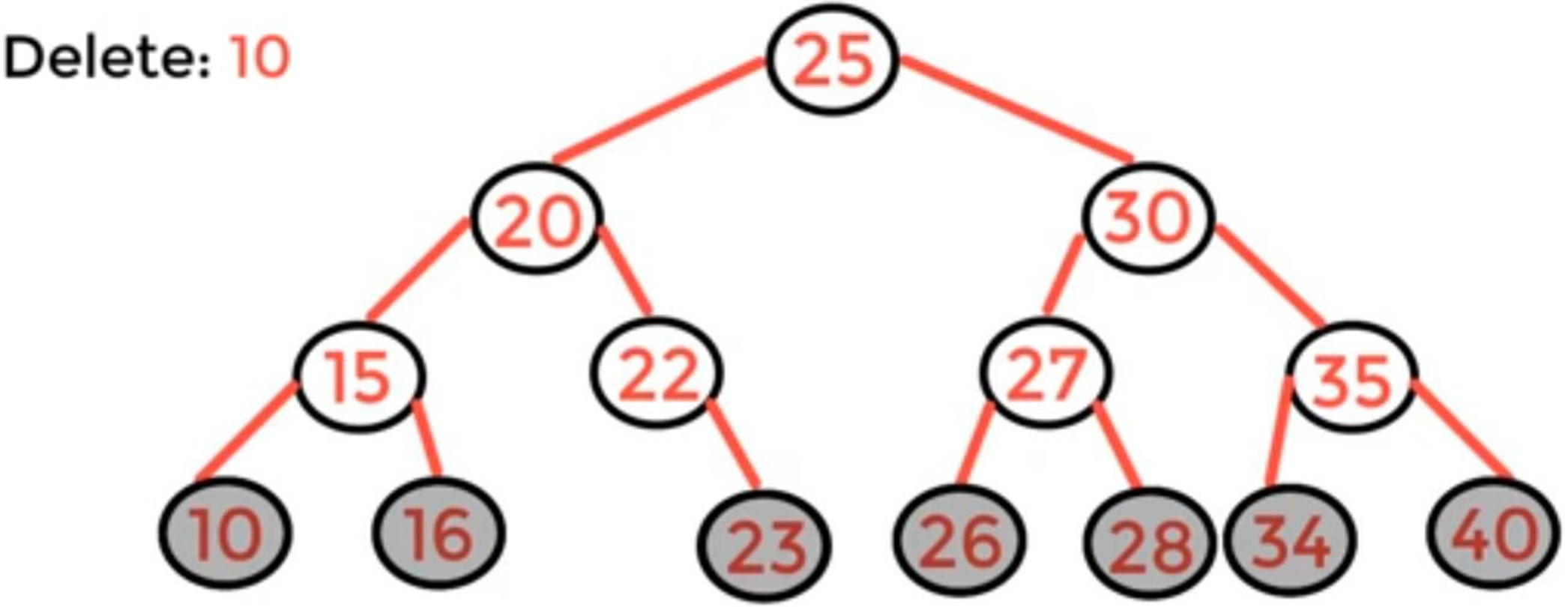


- **مثال 4** : المطلوب بناء شجرة البحث الثنائية لقائمة العناصر التالية:
{ 6 , 4 , 5 , 10 , 2 , 1 , 0 , 11 }

الحذف من شجرة البحث الثنائية

- لحذف عقدة من شجرة بحث ثنائية نميز ثلاث حالات :

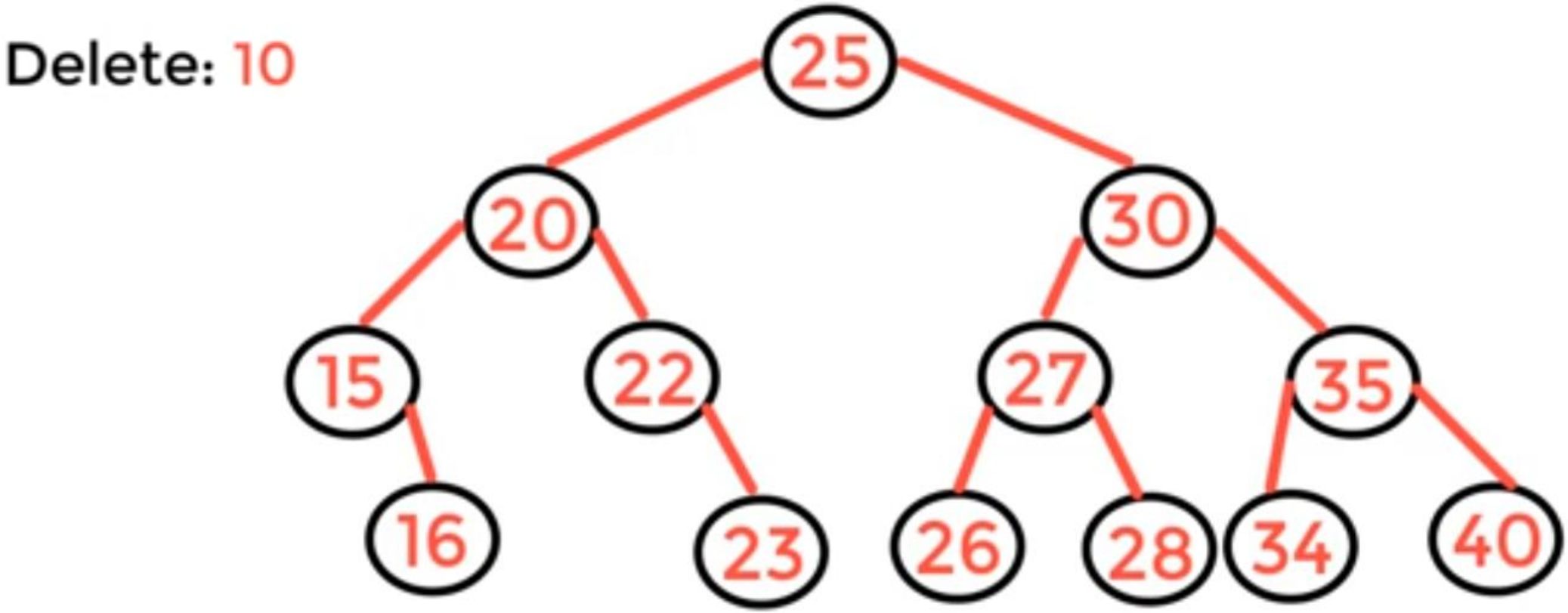
1. إذا كانت العقدة المراد حذفها **عقدة ورقية** ← نجعل حقل الربط للابن اليساري أو اليميني لوالد العقدة يساوي null ثم نحذف هذه العقدة.



الحذف من شجرة البحث الثنائية

- لحذف عقدة من شجرة بحث ثنائية نميز ثلاث حالات :

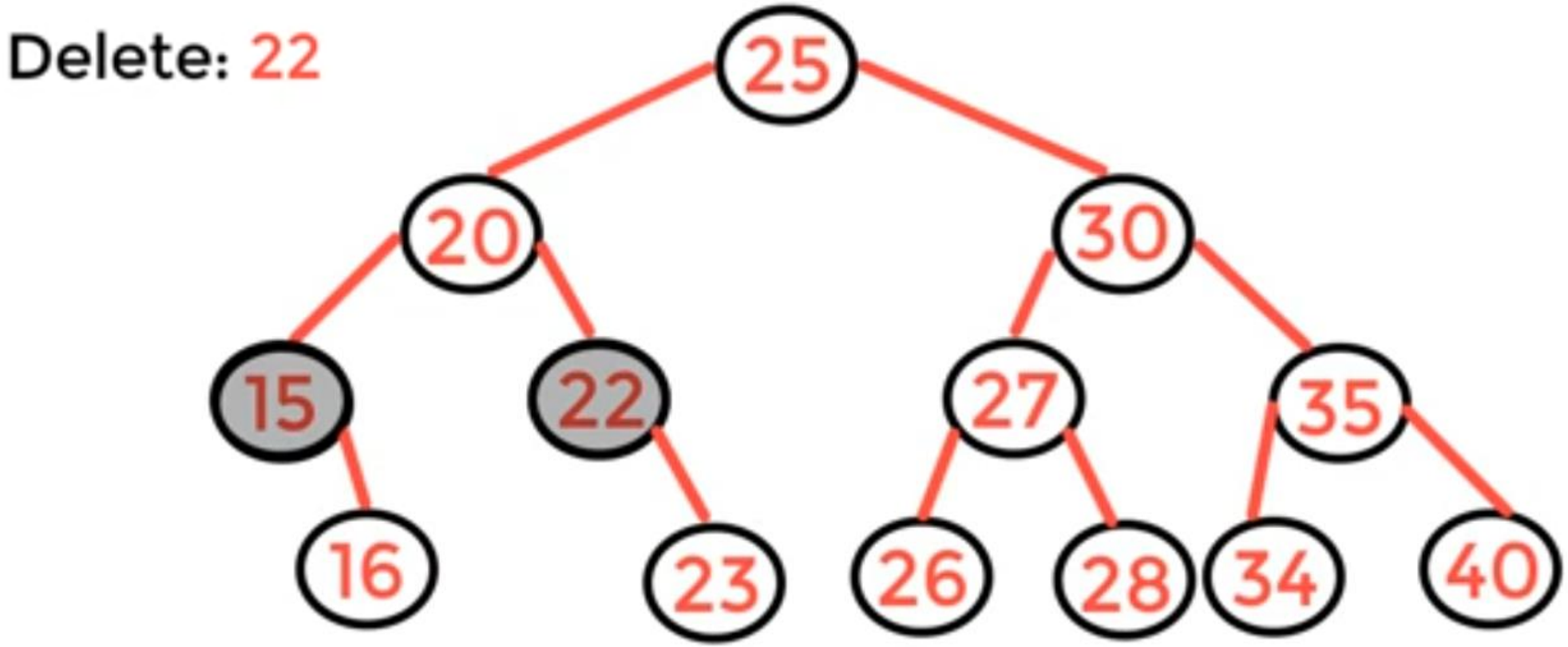
1. إذا كانت العقدة المراد حذفها **عقدة ورقية** ← نجعل حقل الربط للابن اليساري أو اليميني لوالد العقدة يساوي null ثم نحذف هذه العقدة.



الحذف من شجرة البحث الثنائية

- لحذف عقدة من شجرة بحث ثنائية نميز ثلاث حالات :

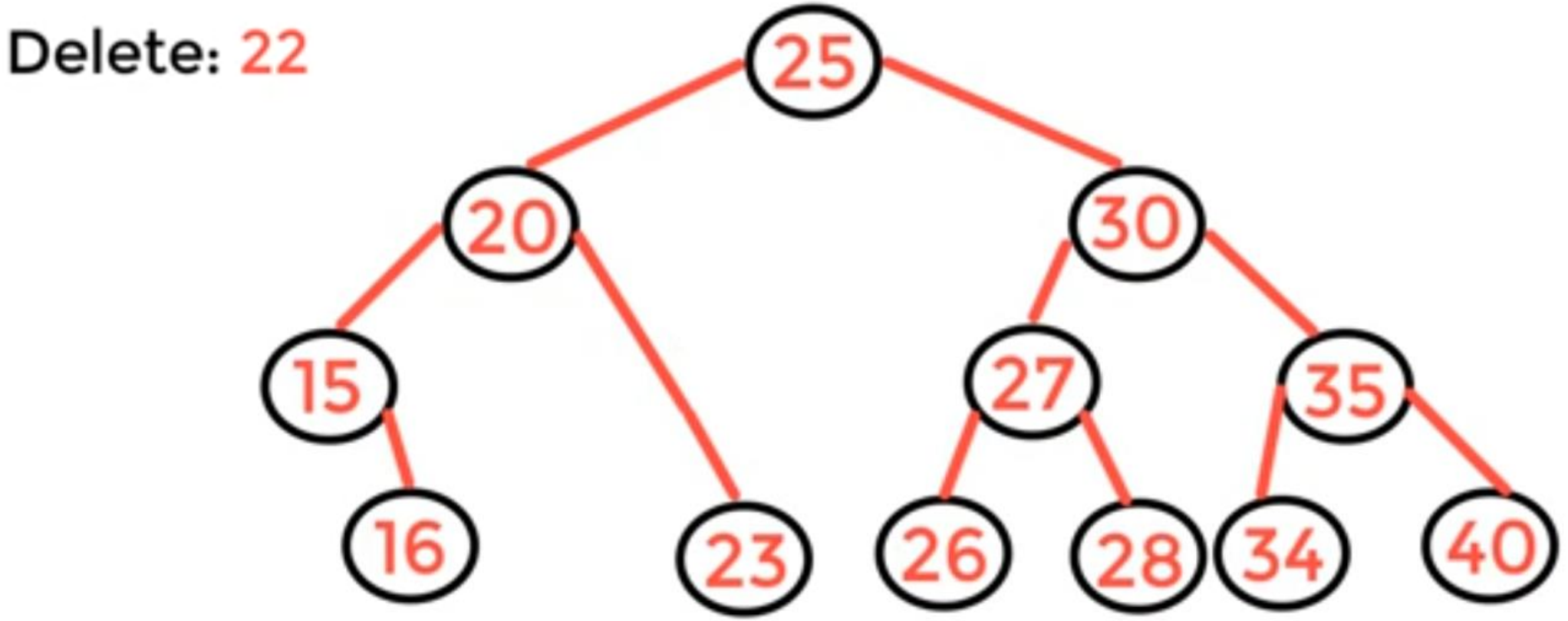
2. إذا كانت العقدة المراد حذفها **عقدة داخلية** لها ابن واحد ← نستبدل هذه العقدة بعقدة الابن .



الحذف من شجرة البحث الثنائية

- لحذف عقدة من شجرة بحث ثنائية نميز ثلاث حالات :

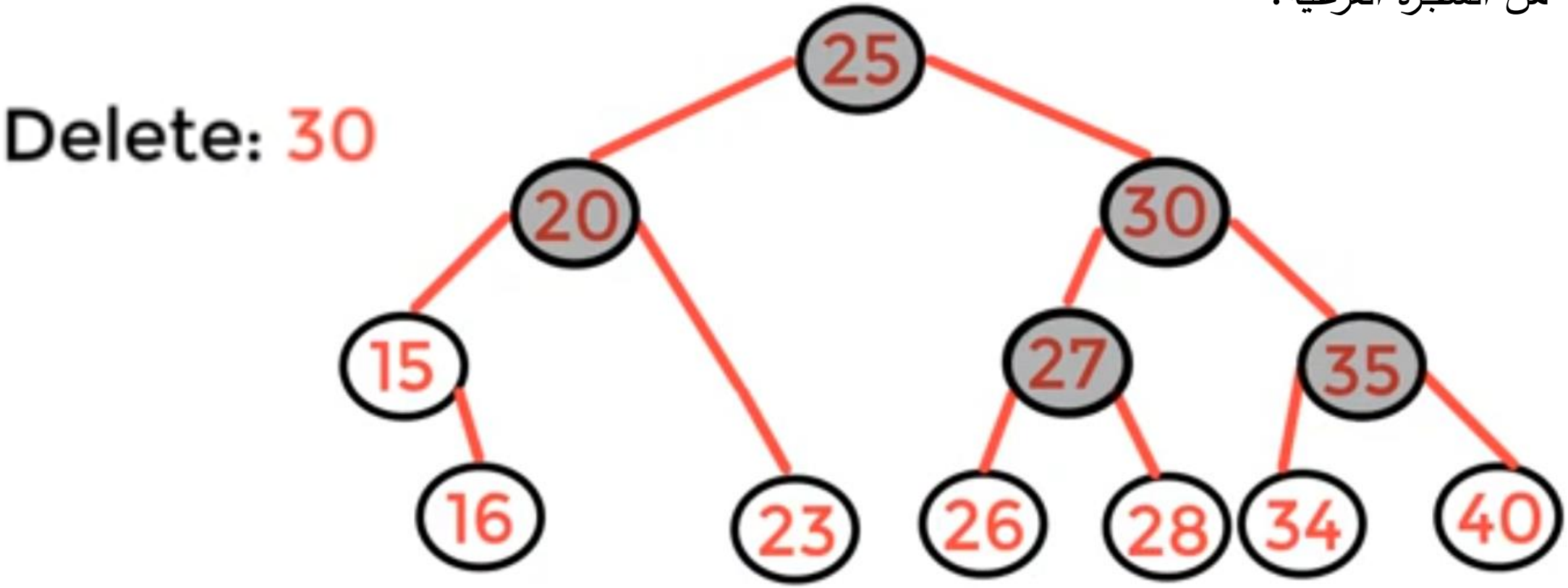
2. إذا كانت العقدة المراد حذفها **عقدة داخلية** لها ابن واحد ← نستبدل هذه العقدة بعقدة الابن .



الحذف من شجرة البحث الثنائية

- لحذف عقدة من شجرة بحث ثنائية نميز ثلاث حالات :

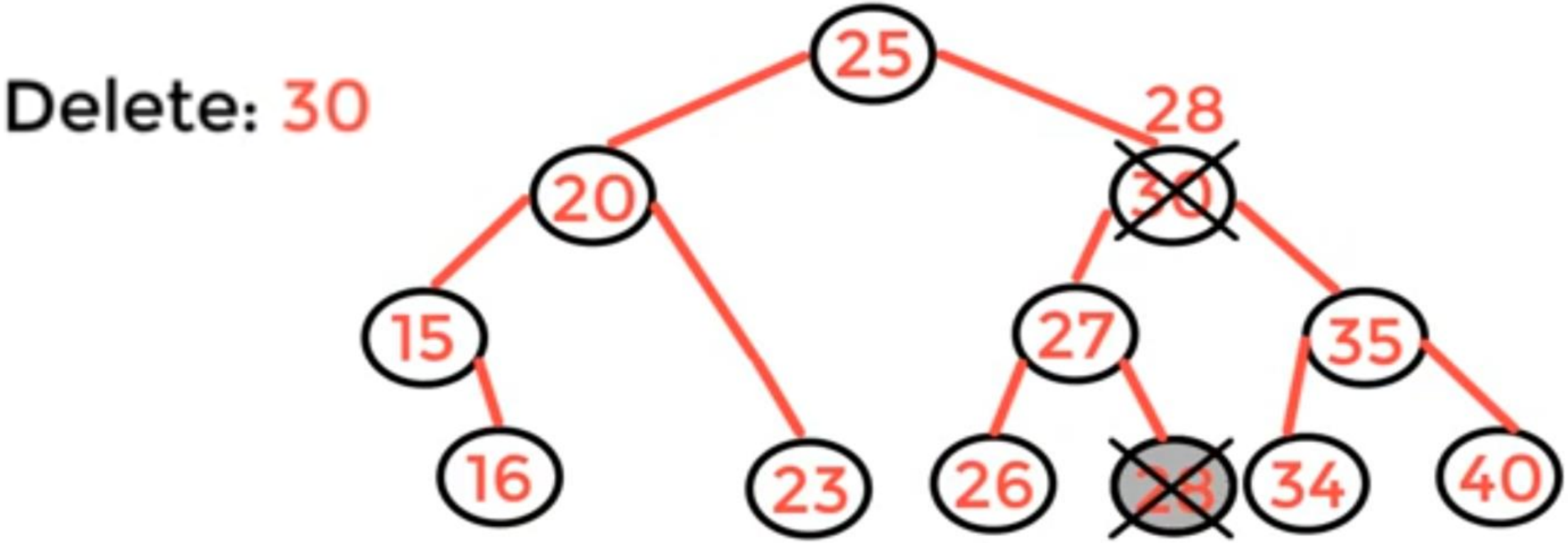
3. إذا كانت العقدة المراد حذفها **عقدة داخلية** لها ولدين← نستبدل هذه العقدة إما بأكبر عقدة في شجرتها الفرعية اليسارية، أو بأصغر عقدة في شجرتها الفرعية اليمينية ومن ثم نباشر بحذف العقدة المستبدلة من الشجرة الفرعية.



الحذف من شجرة البحث الثنائية

- لحذف عقدة من شجرة بحث ثنائية نميز ثلاث حالات :

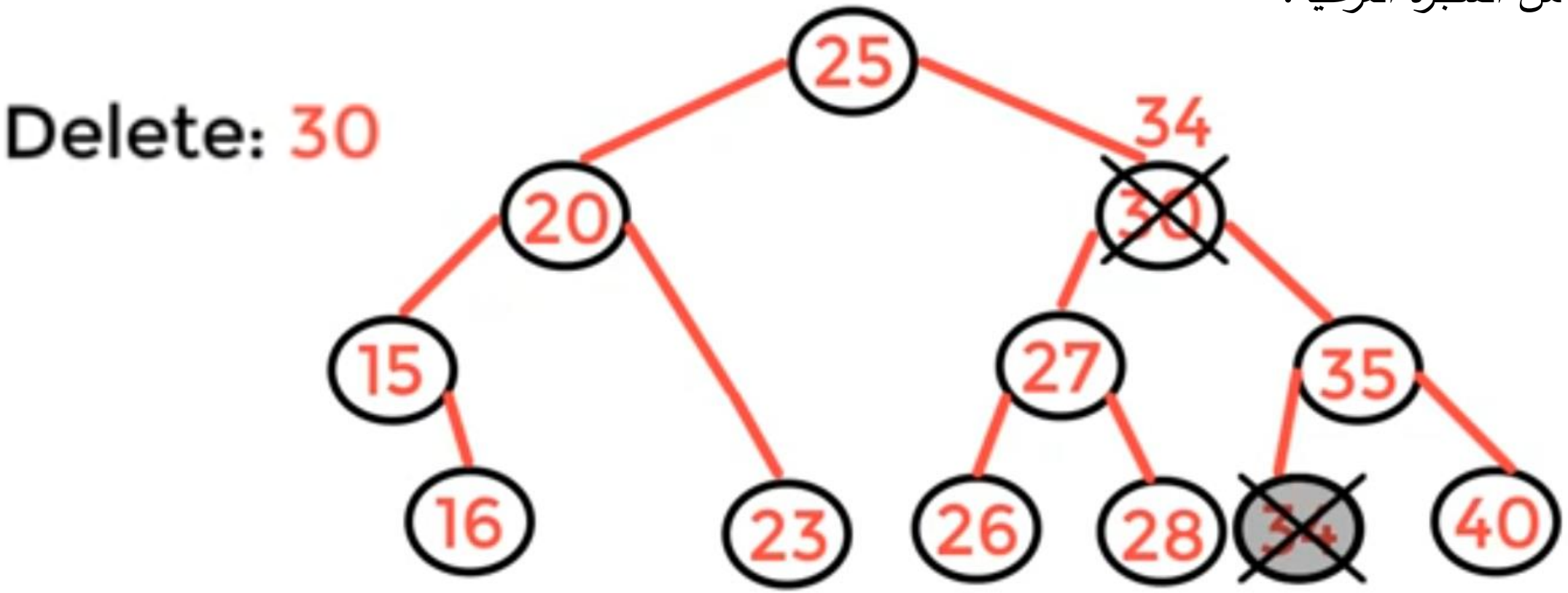
3. إذا كانت العقدة المراد حذفها **عقدة داخلية** لها ولدين← نستبدل هذه العقدة إما بأكبر عقدة في شجرتها الفرعية اليسارية، أو بأصغر عقدة في شجرتها الفرعية اليمينية ومن ثم نباشر بحذف العقدة المستبدلة من الشجرة الفرعية.



الحذف من شجرة البحث الثنائية

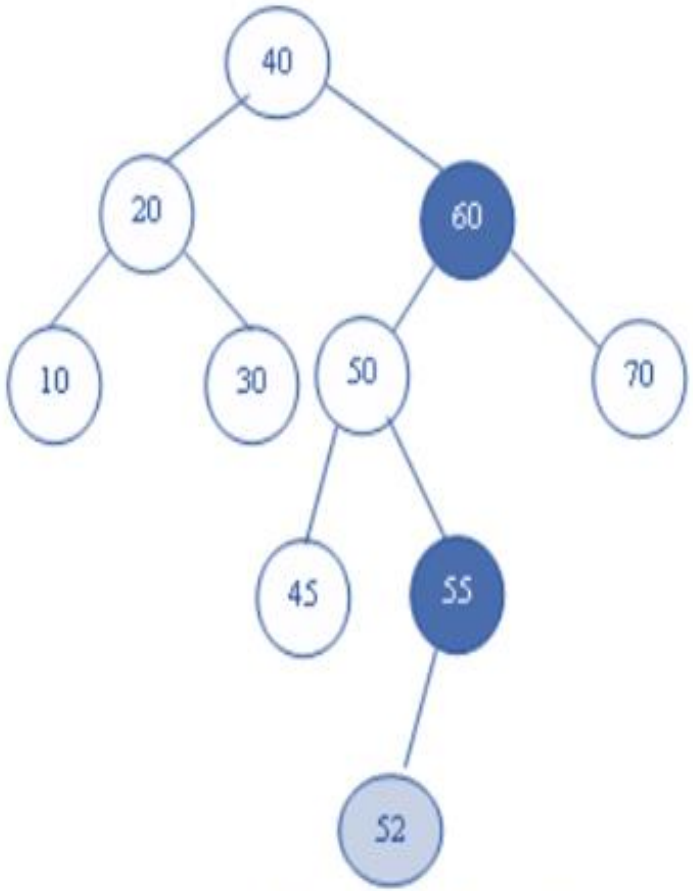
- لحذف عقدة من شجرة بحث ثنائية نميز ثلاث حالات :

3. إذا كانت العقدة المراد حذفها **عقدة داخلية** لها ولدين← نستبدل هذه العقدة إما بأكبر عقدة في شجرتها الفرعية اليسارية، أو بأصغر عقدة في شجرتها الفرعية اليمينية ومن ثم نباشر بحذف العقدة المستبدلة من الشجرة الفرعية.

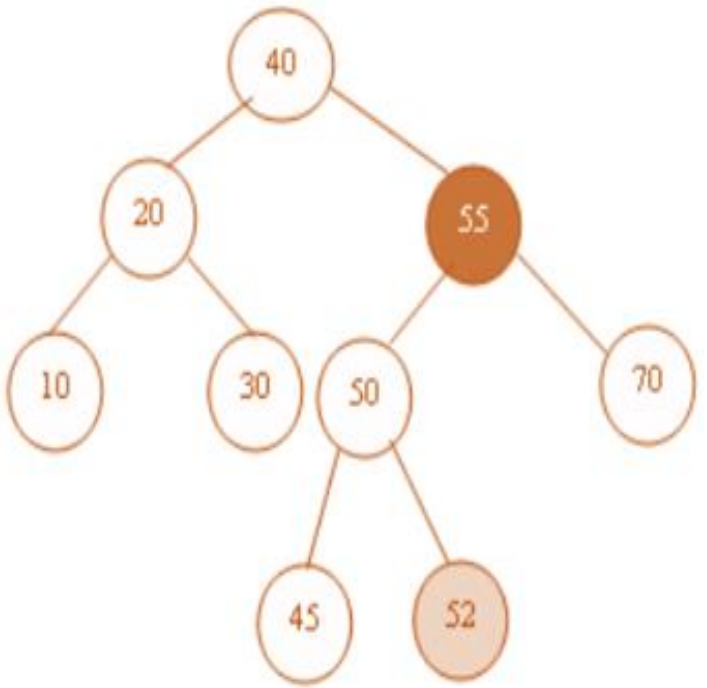


الحذف من شجرة البحث الثنائية

- مثال:



(a) الشجرة قبل حذف 60



(b) الشجرة بعد حذف 60

الحذف من شجرة البحث الثنائية

- الدالة **الإجرائية العودية** المسؤولة عن تنفيذ عملية الحذف هي:

```
/* Given a binary search tree and a key value x, this function deletes the node with
key value x and returns the new root */

node* deleteNode(node* root, int x)
   {    if (root == NULL)  return root;
     // If the key to be deleted is smaller than the root's key, then it lies in left subtree
        if (x < root->key)
           root->left = deleteNode(root->left, x);
     // If the key to be deleted is greater than the root's key, then it lies in right subtree
     else if (x > root->key)
          root->right = deleteNode(root->right, x);

     // if x is same as root's key, then This is the node  to be deleted
     else {
         // node has no child
         if (root->left==NULL and root->right==NULL)
             return NULL;
```

الحذف من شجرة البحث الثنائية

- الدالة **الإجرائية العودية** المسؤولة عن تنفيذ عملية الحذف هي:

```
    // node with only one child
    else if (root->left == NULL) {
        node* temp = root->right;
        free(root);
        return temp;   }
    else if (root->right == NULL) {
        node* temp = root->left;
        free(root);
        return temp;        }
    // node with two children: Get the inorder successor
   // (smallest in the right subtree)
    node* temp = minimum(root->right);
   // Copy the inorder successor's content
       int val = temp->key;
   // Delete the inorder successor
    root->right = deleteNode(root->right, temp->key);
    root->key=val;
}
return root;}
```